Analysis III WS07, Aufgabenblatt 9 **M. Hortmann**				
Name(n)				***Gruppennummer***
Punkte				
1	2	3	Summe	% bearbeitet

Differentialgleichungen ohne Computer: das macht nicht viel Sinn.

Die Bearbeitung der folgenden 3 Aufgaben ist zum Teil recht aufwändig.
Punkte gibt es bereits für eine angemessene ernsthafte Dokumentation ihrer (versuchten) Lösungswege. Bei den Graphikaufgaben steht es Ihnen natürlich frei, andere Software Ihrer Wahl zu verwenden. Sie sollen Ihr Vorgehen aber jedenfalls exakt beschreiben.

Aufgabe 1

Sei $A=\begin{pmatrix} 0 & -67 & -51 & -14 & -26 & 57 & 133 \\ 11 & -251 & -207 & -8 & -59 & 193 & 394 \\ -7 & 253 & 212 & 0 & 68 & -199 & -405 \\ 1 & 57 & 47 & 4 & 20 & -48 & -101 \\ 11 & -47 & -39 & 2 & 10 & 24 & 35 \\ 2 & 21 & 22 & -13 & 12 & -19 & -36 \\ 4 & -44 & -37 & 2 & -6 & 31 & 60 \end{pmatrix}$.

Berechnen Sie mit Hilfe von Pari-gp eine Fundamentallösung $\Phi(x)$ des homogenen Differentialgleichungssystems $y'=Ay$ mit $\Phi(0)=E$.
Dokumentieren Sie jeden benutzen Pari-Befehl und dessen Output.
(Es ist auch für Word oder OpenOffice hilfreich, zur Output-Formatierung den Befehl printtex zu einzusetzen.)

Vorbemerkungen zu graphischen Darstellungen

Graphik mit Pari

Informieren Sie sich über die Graphikfähigkeiten von Pari im Abschnitt Graphik der Pari-Reference-Card „refcard.pdf“, sowie in den entsprechenden Abschnitten des Pari-User-Manuals und des Tutorials, welche der Pari-Distribution beiliegen. Achten Sie darauf, daß Sie eine Pari-Version ab 2.2. einsetzen.

Wie wir schon wissen, sind die Stromlinien des Vektorfelds $X(x,y)=(-y,x)$ Kreise. Nach Eingabe der Pari-Codezeilen

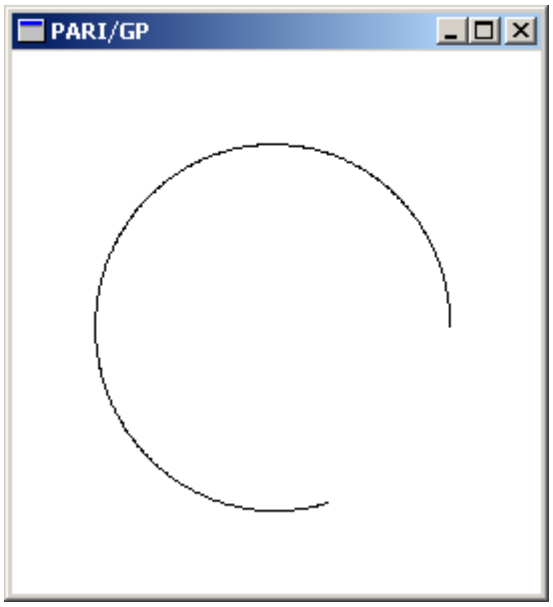
```
euler(x,X,epsilon)=return(x+epsilon*X)

X1(x)=return([-x[2],x[1]])
c=[1,0]
epsilon=0.01

plotinit(2)
plotscale(2,-1.5,1.5,-1.5,1.5)
plotmove(2,1,0)
for(i=0,500,c=euler(c,X1(c),epsilon);plotlines(2,c[1],c[2]);)

plotdraw([2,-2,2])
```

erscheint folgendes Graphikfenster:



welches mit einem Screenshot-Tool kopiert und an dieser Stelle eingefügt wurde.

Analyse der obigen Programmzeilen:
Mit „plotinit(2)“ wird ein Fenster mit der Kennummer 2 initialisiert,
„plotscale(2,-1.5,1.5,-1.5,1.5)“ setzt den Fensterausschnitt von Fenster 2 auf das Quadrat $[-1.5,1.5]\times[-1.5,1.5]$.
„plotmove(2,1,0)“ setzt den Cursor im Fenster 2 auf den Punkt (1,0).
„plotlines(2,c[1],c[2])“ zieht eine Strecke vom Cursorpunkt zum Punkt c=(c[1],c[2]) und setzt den Cursor anschließend auf den Endpunkt der Strecke.
Mit dem Befehl „plotdraw([2,-2,2]“ wird schließlich das Fenster 2 tatsächlich auf dem Bildschirm gezeichnet, wobei (-2,2) die obere linke Ecke des Bildschirmfensters bezeichnet.
Pari bietet mit dem Befehl „psdraw“ statt „plotdraw“ die Möglichkeit, den Output statt auf den Bildschirm in eine Postscriptdatei pari.ps zu schreiben. Darüber hinaus gibt es die Möglichkeit, den Output direkt als Input für das Programm Gnuplot zu exportieren.

Installieren Sie jedenfalls auf Ihrem Windows- oder Linux-Rechner das Programm Gnuplot von www.gnuplot.info .

Durch die Pari-Codezeilen

```
euler(x,X,epsilon)=return(x+epsilon*X)

X1(x)=return([-x[2],x[1]])
c=[1,0]
```

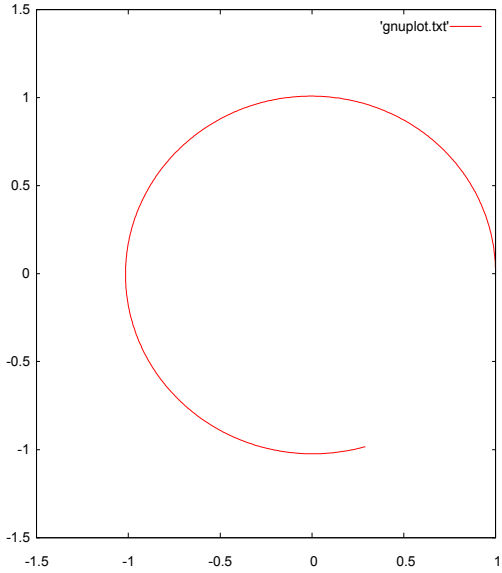
```
epsilon=0.01

for(i=0,500,write("gnuplot.txt",c[1]," ",c[2]);c=euler(c,X1(c),epsilon))
```

wird eine Datei namens „gnuplot.txt“ erzeugt, deren erste Zeilen so aussehen:

```
1 0
1 0.010000000000000000000000000000
0.999900000000000000000000000000 0.020000000000000000000000000000
0.999700000000000000000000000000 0.029999000000000000000000000000
0.999400010000000000000000000000 0.039996000000000000000000000000
0.999000050000000000000000000000 0.049990000100000000000000000000
0.998500149999000000000000000000 0.059980000600000000000000000000
0.997900349993000000000000000000 0.069965002099990000000000000000
0.997200699972000100000000000000 0.079944005599920000000000000000
0.996401259916000900000000000000 0.089916012599640001000000000000
0.995502099790004499990000000000 0.099880025198800010000000000000
```

Wenn man nun in gnuplot den Befehl „plot 'gnuplot.txt' “ gibt, wird folgendes Bild angezeigt.



Klickt man mit der rechten Maustaste auf die zugehörige Fenster-Titel-Zeile, so kann mit dem sich anbietenden Kontext-Menü das Bild in die Zwischenablage und von dort in ein Text-Dokument kopieren, z.B. in Word oder wie hier in Open-Office (www.openoffice.org) .

Achten Sie noch darauf, daß Sie innerhalb Gnuplot mit dem Menübutton „ChDir“ ein Verzeichnis einstellen, in dem die mit dem plot-Befehl referenzierten Dateien anzufinden sind, und geben Sie ggf. den Befehl „set style data lines“, damit nicht Punkte, sondern Linien geplottet werden. Letztlich können Sie aber auch mit den Pari-eigenen Plotbefehlen „ploth“ etc. experimentieren und sich im Pari-Manual „users.pdf“ oder im Tutorial „tutorial.pdf“ dazu informieren

Studieren Sie auch die den Demoprogrammen im Gnuplot-Paket beiliegende Datei „vector.dem“ und führen Sie sie mit dem Befehl „load vector.dem“ aus, wobei natürlich mit „ChDir“ das entsprechende Verzeichnis eingestellt sein muß. Sie lernen dabei, mit Hilfe von Gnuplot Vektorfelder graphisch darzustellen.

Zur Berechnung von numerischen Lösungen gewöhnlicher Differentialgleichungssysteme bzw. zur numerischen Berechnung von Stromlinien von Vektorfeldern eignet sich das in den Gnu-Plot-Utilities enthaltene Programm „ode“. Ode erzeugt Output, den Sie wieder in Gnuplot verwenden können. Download unter http://www.gnu.org/software/plotutils oder für Windows unter gnuwin32.sourceforge.net/packages/plotutils.htm. Beachten Sie, daß Sie auf der Kommandozeile von ode für Windows am Zeilenende nicht „return“ drücken dürfen, sondern über den Zifferntastenblock der Tastatur „alt 1 0“ das Unix-Zeilenende-Zeichen direkt eingeben müssen.

Auch dürfen in Dateien mit Befehlen für ode für Windows keine DOS-Zeilenende Zeichen mit Ascii-Code 13 auftauchen, sondern nur die Unix-Zeilenenden mit Ascii-Code 10. Insbesondere das zum Paket gehörende Demo-Programm „orbit.ode“, welches die Bahnkurve eines Planeten in einem Doppelsonnensystem berechnet, ist instruktiv.

Wie man mit Vektorfeldern und gewöhnlichen Differentialgleichungssystemen und ihrer Visualisierung in Maple umgeht, können Maple-Benutzer anhand der Maple-Arbeitsblätter wise.fau.edu/~ebelogay/teach/2302/sysdeplot.mws bzw. wise.fau.edu/~ebelogay/teach/2302/dfieldplot0.mws studieren.

Aufgabe 2

a) Führen Sie die oben beschriebene Iteration
`for(i=0,500,c=euler(c,X1(c),epsilon);plotlines(2,c[1],c[2]);)`
5000 mal durch und stellen Sie das Ergebnis wie oben graphisch dar.
Wählen Sie ein kleineres epsilon und vergrößern die Anzahl der Schleifendurchläufe entsprechend. Konsequenz?

b) Das Runge Kutta-Verfahren verfeinert das Euler-Verfahren so:
Ist X ein Vektorfeld im $\mathbb{R}^n$, $x_0 \in \mathbb{R}^n$ ein Startwert, $\epsilon > 0$, so definiere man x_{n+1} nicht wie beim Eulerverfahren durch $x_{n+1} = x_n + \epsilon X(x_n)$, sondern, indem man zunächst setzt

$$x_A = x_n, X_A = X(x_A) \ ,$$

$$x_B = x_A + \frac{\epsilon}{2} X_A, X_B = X(x_B) \ ,$$

$$x_C = x_A + \frac{\epsilon}{2} X_B, x_C = X(x_C) \ ,$$

$$x_D = x_A + \epsilon X_C, X_D = X(x_D)$$

durch

$$x_{n+1} = x_n + \epsilon \frac{1}{6}\left(X_A + 2X_B + 2X_C + X_D\right)$$

Die Lösungskurve $c(t)$ mit $c(0) = x_0$ wird dann approximiert durch $c(n\epsilon) = x_n$.

Statt des durch `euler(x,X,epsilon)=return(x+epsilon*X)` gegebenen Eulerverfahrens definieren Sie diesmal eine Funktion `runge-kutta(x,X,epsilon)` und erzeugen damit den analogen Graphik-Output wie in a) , so daß man anhand dieses Beispiels beide Verfahren vergleichen kann.

Aufgabe 3

Nach dem Newtonschen Gravitationsgesetz ist die Beschleunigung, die die Erde in Richtung Sonne bzw. einem anderen Planeten erfährt, proportional zum jeweils Inversen des Abstandsquadrats.

Wir denken uns den $\mathbb{R}^2$ mit der Sonne im Nullpunkt und zwei Planeten in Umlaufbahnen, Zur Berechnung der Bahnen berücksichtigen wir die Anziehung der Sonne auf beide Planeten und die 100 mal schwächer angenommen Anziehung der beiden Planeten zueinander. Wir berücksichtigen der Einfachheit halber nicht die Kräfte, die die Planeten auf die Sonne ausüben. Die Kräfte sind lt. Newtonschem Gravitationsgesetz auf den jeweiligen Anziehungspartner gerichtet und proportional zum inversen Quadrat der Entfernung.

Um dies zu modellieren, nehmen wir einen $\mathbb{R}^8$ mit den Koordinaten $(x_1, y_1, vx_1, vy_1, x_2, y_2, vx_2, vy_2)$. Dabei müßten noch die Punkte ausgenommen werden, in denen die Planeten miteinander oder mit der Sonne kollidieren; es darf also nicht $x_1 = x_2$ und $y_1 = y_2$ oder $x_1 = 0$ und $y_1 = 0$ oder $x_2 = 0$ und $y_2 = 0$ gelten, d.h. wir betrachten eigentlich nur die offene Teilmenge $U \subset \mathbb{R}^8$, die diese Kollisionen nicht enthält.

Jetzt haben wir folgendes Differentialgleichungssystem

$$x_1' = vx_1,\ y_1' = vy_1$$

$$vx_1' = \frac{-x_1}{\|(x_1, y_1)\|^3} + \frac{1}{100}\frac{x_2 - x_1}{\|(x_2 - x_1, y_2 - y_1)\|^3}$$

$$vy_1' = \frac{-y_1}{\|(x_1, y_1)\|^3} + \frac{1}{100}\frac{y_2 - y_1}{\|(x_2 - x_1, y_2 - y_1)\|^3}$$

$$x_2' = vx_2,\ y_2' = vy_2$$

$$vx_2' = \frac{-x_2}{\|(x_2, y_2)\|^3} + \frac{1}{100}\frac{x_1 - x_2}{\|(x_2 - x_1, y_2 - y_1)\|^3}$$

$$vy_2' = \frac{-y_2}{\|(x_2, y_2)\|^3} + \frac{1}{100}\frac{y_1 - y_2}{\|(x_2 - x_1, y_2 - y_1)\|^3}$$

Es sind im folgenden verschiedene Lösungen des Systems graphisch darzustellen. Das zugehörige Vektorfeld ist offenbar gegeben durch
$X(x_1, y_1, vx_1, vy_1, x_2, y_2, vx_2, vy_2) = (x_1', y_1', vx_1', vy_1', x_2', y_2', vx_2', vy_2')$

Sie sollen Anfangsbedingungen finden, bei denen

a) die Planeten sich gegenseitig umkreisend um die Sonne drehen,
b) bei denen sie zwei unabhängige Bahnen um die Sonne einnehmen, und
c) bei denen das System chaotisch oder instabil wird.

Beim numerischen Rechnen mit Euler oder Runge-Kutta ergibt sich eine Folge von Punkten $(x_1, y_1, vx_1, vy_1, x_2, y_2, vx_2, vy_2)$. Die gesuchten Planetenbahnen sind offenbar gegeben durch die Liste der (x_1, y_1) einerseits und der (x_2, y_2) andererseits, und nur diese sind zu plotten.

Bemerkung:
Eine mögliche Verallgemeinerung der obigen Aufgabenstellung wäre z.B. ein gravitatives n-Körper Szenario. Man würde *n* Punkte im dreidimensionalen Raum verteilen und ihnen Massen und Anfangsimpulse mitgeben. Eine Simulation sollte dann die Bahnkurven in Echtzeit gemeinsam plotten, wobei die Zeit dehnbar und streckbar sein sollte. Die Kurven müßten auch nach einer wählbaren Zeit wieder gelöscht werden können, damit sie den Raum nicht „vollmüllen“. Auch sollten Kollisionen und Konstellationen vorausberechnet werden können.